N80-21375

Analysis of a Class of Totally Self-Checking Circuits Implemented in an NMOS Custom LSI Structure

M. W. Sievers
Communications Systems Research Section

In the design of ultrareliable computer systems, circuitry must be provided to initiate and execute error control procedures. These procedures include system recovery, data error detection and correction, and handling of singular and exceptional conditions. These procedures are very often implemented by circuitry whose inputs remain constant until an error condition occurs. These circuits, therefore, cannot be fully tested during normal system operation. Lurking faults may be present that will mask the effects of faults in the circuitry being checked

It is possible to replace constant valued signal lines with a pair of signal lines such that a given logic state has a redundant representation. A class of Boolean algebra called morphic Boolean algebra can then be used to realize totally self-checking circuits to replace semi-passive error handling circuitry. The fault detection properties of these circuits under the assumption of classical stuck-at faults have already been investigated. The purpose of this paper is to specify the expected nonclassical faults in a particular NMOS custom structure and to predict their effect on morphic circuits.

I. Introduction

Fault-tolerant computer systems are made possible only when errors are detected and recovery procedures initiated before permanent data contamination occurs. An error is defined as the corruption of any single line logic state.

Errors may appear anywhere in the implementation of a logic function. A single circuit fault may cause zero, one, or more errors depending on the function, the fault, and the input pattern. A particular line in the implementation of a function is testable if and only if for a given input pattern, a change in the logic state of the line causes a change in the output function.

Lines are termed active if their logic state changes during normal operation. Lines whose state remains constant during normal operation are called semi-passive. The difficulty in detecting errors in active lines is that they are always intermittent. Semi-passive lines are frequently used in error recovery circuitry and do not change state unless an exceptional condition occurs. The error recovery circuitry, therefore, cannot be fully tested. Lurking faults may be present that may mask errors in active lines, causing irrepairable data damage.

A means of checking the working condition of error recovery circuitry is to implement them in a totally self-checking scheme. These schemes require that input and output

data be coded so that it is possible to distinguish between valid and invalid codewords. Totally self-checking has been defined by Anderson (Ref. 1) as follows:

Definition 1:

A circuit is termed *self-testing* if for every fault in a given set, the circuit will produce an invalid codeword for at least one valid input codeword.

Definition 2:

A circuit is *fault secure* if for every fault in a given set, the circuit either generates an invalid output codeword or yields the same codeword as a fault-free circuit.

Definition 3:

A *totally self-checking* circuit is both self-testing and fault secure.

Total self-checking requires that input variables change state in such a way that all paths through the circuit can be sensitized. This requirement is incompatible with semi-passive signal lines. The necessary condition that logic lines change state requires that single semi-passive lines be replaced by a pair of lines, both of which switch state in normal operation. Carter (Refs. 2 and 3), has reported on a mapping from semi-passive lines to pairs of lines such that both states have redundant representations. Morphic Boolean algebra operating on these line pairs can realize totally self-checking circuits under the assumption of single stuck-at faults.

The purpose of this article is to investigate the effects of nonclassical faults, predicted for a particular NMOS circuit realization, on the self-checking properties of morphic logic. Faults examined will be those either present at the manufacturing time of a chip or those that appear later due to wear-out.

The next section will present the formalities of morphic logic. Section III offers a failure model for the predicted NMOS failures in a structure for custom LSI. The last section will deal with the expected effect of nonclassical faults on morphic self-checking.

II. Morphic Boolean Algebra

A mapping from single semi-passive signal lines to a pair of lines is defined as follows:

$$M: \quad [(e_1, e_2), (\bar{e}_1, \bar{e}_2)] \rightarrow 1$$

$$[(\bar{e}_1, e_2), (e_1, \bar{e}_2)] \rightarrow 0$$

where $e_1, e_2 \in \{0, 1\}$ (Ref. 2). Each wire in the pair can take both values of 0 and 1 without changing the logic state associated with a single line logic. The equivalent single line state is easily determined by the parity of the pair of lines.

Two separate problems arise. The first question is one of defining a correspondence between a function g in ordinary Boolean algebra $B = \{0, 1, \{*\}\}$ where $\{*\}$ is the usual set of logic operators, and a function G whose inputs and outputs satisfy the mapping M. It is necessary to define Boolean operators over the pair $\{e_1, e_2\}$. This is done by expanding the mapping M to be a *morphism* between B and

$$\{\{(e_1, e_2), (\bar{e}_1, \bar{e}_2)\}, \{(\bar{e}_1, e_2), (e_1, \bar{e}_2)\}, \{*m\}\}$$

where $*m$ is defined as the set of morphic Boolean operators.

Definition 4:

A *morphism* is a general mapping from a set A into a set B that preserves the algebraic structure of A.

Let

$$s_i, s_j \in \{(0,0), (0,1), (1,0), (1,1)\},$$

then

$$M(s_i *m\, s_j) = M(s_i) * M(s_j)$$

defines the set of morphic Boolean operators $*m$, and

$$B_m = \{\{(e_1,e_2), (e_1,e_2)\}, \{(e_1,e_2), (e_1,\bar{e}_2)\}, \{*m\}\}$$

forms a Boolean algebra by the definition of morphism M.

It was shown above that semi-passive signal lines could be replaced by a pair of lines and that operators could be defined for the pair. For n input bits in the semi-passive function, there are 2^{2^n} possible morphic Boolean functions. Not all of these will be capable of testing all n input pairs. The second question to be answered is how to determine the self-checking and self-testing properties of the circuit. The following assumes stuck-at faults.

Definition 5:

An input A_i of a morphic Boolean function G and an input pattern p is *testable* when a variation in A_i causes a variation in G.

Definition 6:

The *Boolean difference* of a morphic function G with respect to an input A_i is defined as

$$\Delta_{A_i} G(A_1, A_2, \ldots, A_i, \ldots, A_n) =$$

$$G(A_1, \ldots, A_i, \ldots, A_n) \oplus G(A_1, A_2, \ldots, \overline{A_i}, \ldots A_n)$$

where $\oplus$ is the EXCLUSIVE OR function.

It is easily shown that $\Delta_{A_i} G$ is a function of $A_1, A_2, \ldots, A_{i-1}, A_{i+1}, \ldots, A_n$ only. These two definitions lead to the following theorem.

Theorem 1:

An input A_i implemented as (A_{i1}, A_{i2}) of the morphic function $G(A_1, A_2, \ldots, A_i, \ldots, A_n)$ with a given pattern p in the input space is tested if and only if

$$\Delta_{A_i} G \text{ is TRUE.}$$

Proof:

The theorem follows immediately from definitions 5 and 6.

The next theorem shows a simple method for determining the testability of an input to a morphic function solely as a function of the semi-passive input pattern and the functional specification.

Theorem 2:

Let a semi-passive input pattern p be such that A_i is FALSE for $i = 0, 1, 2, \ldots, k$ inputs and A_i is TRUE for the remaining inputs. A term A_j is testable at the output of a function G if and only if the number of terms containing A_j and not A_i, $i \neq j$ and $i = 0, 1, 2, \ldots, k$ (the FALSE terms) in the Boolean polynomial equivalent to G is odd.

Proof:

The theorem is proved by applying Theorem 1:

Let

$$G(A_1, A_2, \ldots, A_n) = C_0 \oplus C_1 A_1 \oplus \ldots, \oplus C_n A_n$$

$$+ C_{n+1} A_1 A_2 \oplus \ldots,$$

$$\oplus C_{2^n - 1} A_1 A_2 \ldots, A_n$$

be the Boolean polynomial equivalent to G, where $C_i \epsilon \{\text{TRUE, FALSE}\}$. Let X_0 equal the sum of terms in the polynomial equivalent to G that do not contain A_j and X_i be the product of $i - 1$ distinct variables A_{i_k}, $A_{i_k} \neq A_j$. The general Boolean polynomial equivalent to G may be rewritten as

$$G(A_1, A_2, \ldots, A_n) = d_0 X_0 \oplus d_1 A_j \oplus d_2 A_j X_2$$

$$\oplus d_n A_j X_n$$

The Boolean difference of G with respect to A_j is

$$\Delta A_j G = d_1 \oplus d_2 X_2 \oplus \ldots, \oplus d_m X_m \oplus \ldots, \oplus d_n X_n$$

All X_m's which contain A_i, $i = 0, 1, \ldots, k$ such that A_i is FALSE are FALSE. Therefore, since the condition of testability of A_j for a pattern p is by Theorem 1

$$\Delta_{A_j}\Big|_p = \text{TRUE}$$

the EXCLUSIVE OR of all d_m's for X's not containing FALSE A_i's along with d_1 must be TRUE. Therefore, the parity of terms containing A_j and not FALSE A_i's must be odd.

The previous two theorems indicate methods for determining the testability of the input space of a given function. It remains to be shown how to realize totally self-checking morphic functions.

Reference 2 presents a universal morphic logic set of totally self-checking building blocks. This set consists of morphic AND, NOT and EXCLUSIVE OR functions (see Fig. 1). Not all combinations of these elements are self-checking. The following theorem states the necessary connection conditions.

Theorem 3:

A totally self-checking morphic circuit can be realized if and only if the implementation does not prohibit the application of all necessary test vectors to any of its morphic building blocks.

Proof:

Fault security is assured since any invalid input code to any of the morphic building blocks will result in an invalid output codeword. The effect of the invalid input propagates to the output.

Self-testing can be shown as follows. The realization of the morphic function permits the application of all test vectors to each building block by assumption. Thus, any single fault in any building block will eventually result in the output of an invalid codeword from the faulty element. The invalid codeword again propagates to the output and is detected.

A construction algorithm is given in (Ref. 2) that guarantees totally self-checking realizations. The algorithm does not produce an optimal design either in performance or number of components. However, it is straightforward to implement and could easily be programmed.

III. Failure Model for an NMOS Custom LSI Structure

Figure 2 illustrates a structure for building custom NMOS-integrated circuits. The structure and some of its properties have been reported on previously (Refs. 4 and 5). Logic gates are created and connected together in the structure by a simple mask-level programming scheme.

Due to the low-level implementation of this structure, certain failure mechanisms become more or less likely than other implementations. One of the primary motivations for the structure was to limit the types of defects that might occur to a well-defined set.

In addition to defects that are created in the manufacturing process, there is a set of so-called wear-out flaws and several random failures that determine chip reliability. These flaws are traceable to specification, environment, or fabrication but are not present until some time after the chip is in use. Wear-out ultimately determines the actual useful lifetime of all integrated circuits; however, random failures often occur well before wear-out, although with very low probability.

In well established processes and technologies, little or no correlation exists between the location of processing defects found on chips of the same wafer or run. There is virtually no correlation of location of defects between wafers of different processing lots. If strong correlations did exist, they would be an indication of a fundamental fabrication failure that would easily be detected. Additionally, fabrication houses have a strong financial interest in correcting the fabrication process.

Integrated circuit defects that determine the infant mortality rate and, if undetected, cause some failures during the useful chip life are considered first. Since fabrication flaws tend to be randomly distributed over the surface of a wafer, a reasonably accurate and mathematically tractable model of their presence is the Poisson distribution,

$$P_k = (DA)^k e^{-DA}/k!$$

where P_k is the probability of k flaws occurring on a chip of area A, and D is the flaw density. The probability of no flaws, therefore, is

$$P_0 = e^{-DA}$$

and the probability of at least one flaw is

$$P_{k\geq 1} = 1 - e^{-DA}$$

Only flaws on active circuit areas need to be considered so A should be scaled by r where r is the ratio of active to total circuit area. For the structure in Fig. 2, r has been estimated at 0.7. Flaw density for a typical MOS process is approximately 13.5 flaws/cm^2 (Ref. 6).

In practice, various processing, electrical, and logical tests are performed on wafers and chips to detect the presence of defects.

Definition 7:

Measured functional yield, ym, is the percentage of chips that pass the manufacturer's screening tests.

Measured functional yield is the sum of two yield terms:

Definition 8:

Actual function yield, y, is the actual yield of good chips.

Definition 9:

Define y_{bg} as the yield of bad chips that test good.

Actual functional yield is computed by

$$y = e^{-Da}$$

where $a = Ar$. The yield of chips with k undetected circuit flaws under a test of coverage c is

$$y_{bg}(k) = P_{k,r}(1 - c)^k$$

So the yield of bad chips testing good is

$$y_{bg} = \sum_{k=1}^{\infty} P_{k,r}(1 - c)^k$$

Substituting for $P_{k,r}$:

$$y_{bg} = \sum_{k=1}^{\infty} \frac{e^{-Da}}{k!} (Da)^k (1 - c)^k$$

$$= e^{-Da} (e^{Da(1-c)} - 1)$$

Therefore, the measured functional yield is computed by

$$y_m = e^{-cDa}$$

Table 1 lists the predicted NMOS flaws that affect the yield of circuits built in the structure of Fig. 2. Two of the indicated defects result in classical stuck-at faults. The remainder can be classified either as bridging faults (shorts) or floating faults. A floating condition arises when: (1) a pull-up resistor does not pull up a gate output for some reason; (2) the output contact is missing; or (3) a wire is broken. Capacitive coupling of floating wires to neighboring lines may cause the defective wire to drift between logic states.

The preceding discussion dealt only with processing-induced defects. In addition to those flaws, there is a pair of NMOS wear-out defects that may occur. Table 2 lists their mechanisms and their most probable effects on logic.

Metal migration is a current-induced movement of metal in a direction perpendicular to current flow. It can be slowed to very low levels by reducing the current density in metal wires. The situation is undetectable until the metal finally breaks.

Contamination of gate oxides by sodium ions in the fabrication process is unavoidable. These ions are mobile and tend to accumulate under active transistor gates at the substrate side of the gate oxide. In enhancement-mode pull-down transistors, the initial effect of these ions is to make the substrate appear doped, similar to depletion-mode transistors. This lowers the transistor's threshold voltage making it more difficult to turn off. In NOR logic, under certain circumstances, the result is an input stuck at one, or equivalently, an output stuck at zero. Eventually, a sufficient number of ions will collect in the gate oxide to cause a low impedance path between the gate and substrate. The application of a logical 1 to the gate of the transistor will now punch a pinhole through the gate oxide creating a gate-to-substrate short. The result is a stuck-at zero on the input wire.

Finally, random failures may be due to wire bond failure, ionizing radiation, excessive temperatures, moisture leakage and electrical overstressing. The effects of bond failure and electrical overstressing are similar to those failures already discussed. The remaining mechanisms cause catastrophic failures which are easily detected.

Figure 3 shows a circuit that models the faults indicated in Tables 1 and 2. Gates labeled with an asterisk are susceptible to classical stuck-at faults. All other gates are assumed to be perfect. The transistor switch models the floating condition. When a 1 is applied to the transistor gate, it passes a signal from source to drain. A 0 at the transistor gate breaks the signal path.

IV. Effects of NMOS Defects on Morphic Self-Checking

It has been shown in Section II that totally self-checking morphic circuits can be designed under the assumption of single stuck-at faults. The NMOS defects listed in Tables 1 and 2, however, include nonclassical defects, i.e., bridging faults and floating faults. It will be shown that morphic remains self-checking even when these faults appear.

Bridging faults are easily detected in morphic circuits because they alter an output function in a deterministic manner. Suppose a bridging fault occurred between one line of an input pair, A_{i1}, and one of the output wires, g_1, of a function G. Let the undamaged morphic function be implemented as

$$g_1 = g_1(A_1, A_2, \ldots, A_i, \ldots, A_n)$$

$$g_2 = g_2(A_1, A_2, \ldots, A_i, \ldots, A_n)$$

The damaged function will be

$$g_1 = A_{i1}$$

$$g_2 = g_2(A_1, A_2, \ldots, A_{i-1}, g_1, A_{i+1}, \ldots, A_n)$$

which is easily detected by input patterns sensitizing input A_i.

More generally, for at least one input pattern, any pair of wires in an undamaged morphic circuit must have different logic states. If this were not the case then the pair of wires could be replaced by a single wire, and a short between them would not cause an error. A short between wire pairs is detected by applying the pattern that would normally place the wires into complementary states.

Floating defects results in transient faults. These are always detectable when the wire has floated to the wrong state.

Should the wire be in the correct state, no error is detected, but neither is this an error condition.

The above discussion leads to the following theorem.

Theorem 4:

The realization of any totally self-checking morphic function in the NMOS structure of Fig. 2 remains totally self-checking in the presence of any single fault.

References

1. Anderson, D. A., G. Metze, "Design of Totally Self-Checking Check Circuits for m out of n Codes," *IEEE Trans. Comp.*, Vol. C-22, No. 3, March 1973, pp. 263-269.
2. Carter, W. C., A. B. Wadia, D. C. Jessep, "Implementation of Checkable Acyclic Automata by Morphic Boolean Functions," *Symp. Comp. and Automata*, Polytechnic Institute of Brooklyn, April 1971, pp. 465-482.
3. Carter, W. C., A. B. Wadia, D. C. Jessep, "Computer Error Control By Testable Morphic Boolean Functions – A Way of Removing Hardcore," *Symp. FTC* 1972, pp. 154-159.
4. Sievers, M. W., "A General Logic Structure for Custom LSI," *The Deep Space Network Progress Report 42-50*, Jet Propulsion Laboratory, Pasadena, California, April 15, 1979, pp. 97-105.
5. Sievers, M. W., "Density and Reliability Predictions for A General Logic Structure for Custom LSI," *The Deep Space Network Progress Report 42-53*, Pasadena, California, October 15, 1979, pp. 66-73.
6. Case, G. R., "Analyses of Actual Fault Mechanisms in CMOS Logic Gates," *13th Design Auto. Conf.*, Palo Alto, California, June 27, 1976, pp. 265-270.

Table 1. Effects of various predicted NMOS defects on the structure of Figure 2

Defect	Comment
1. Missing or defective pull-up, broken wire to pull-up	Float
2. Missing pull-down transistor	Input stuck at 1
3. Missing output contact	Float
4. Gate oxide pinhole	Input stuck at 0
5. Broken wire	Float
6. Bridging	Replace shorted terms with their AND

Table 2. Wear out mechanisms

Mechanism	Comment
1. Metal Migration	Float
2. Sodium ion contamination	Initially, input stuck at 1 (equivalent to output stuck at 0); finally input stuck at zero signal from source to drain.

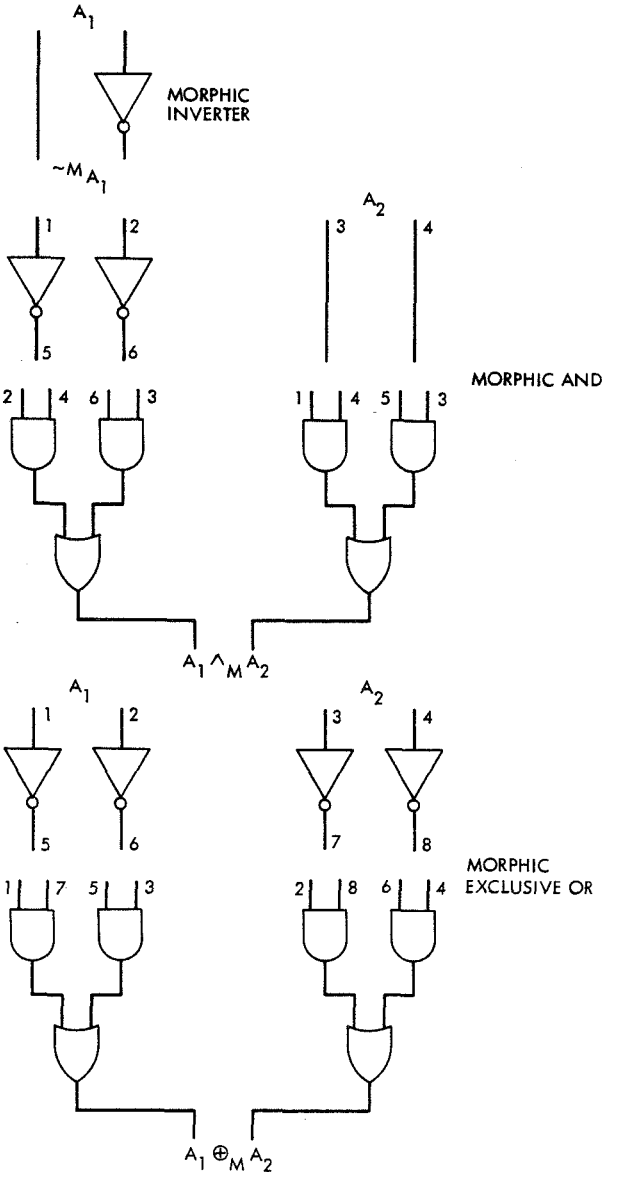

Fig. 1. Morphic building blocks

ORIGINAL PAGE IS
OF POOR QUALITY

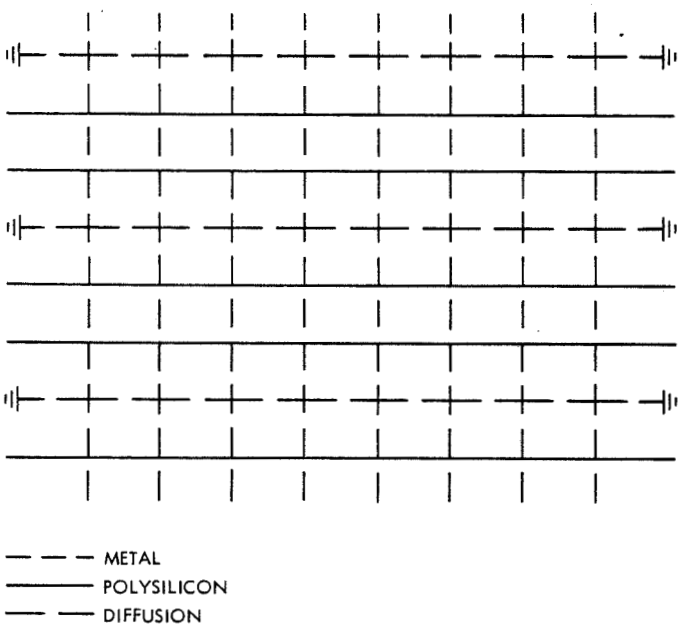

Fig. 2. NMOS general logic structure for integrated circuits

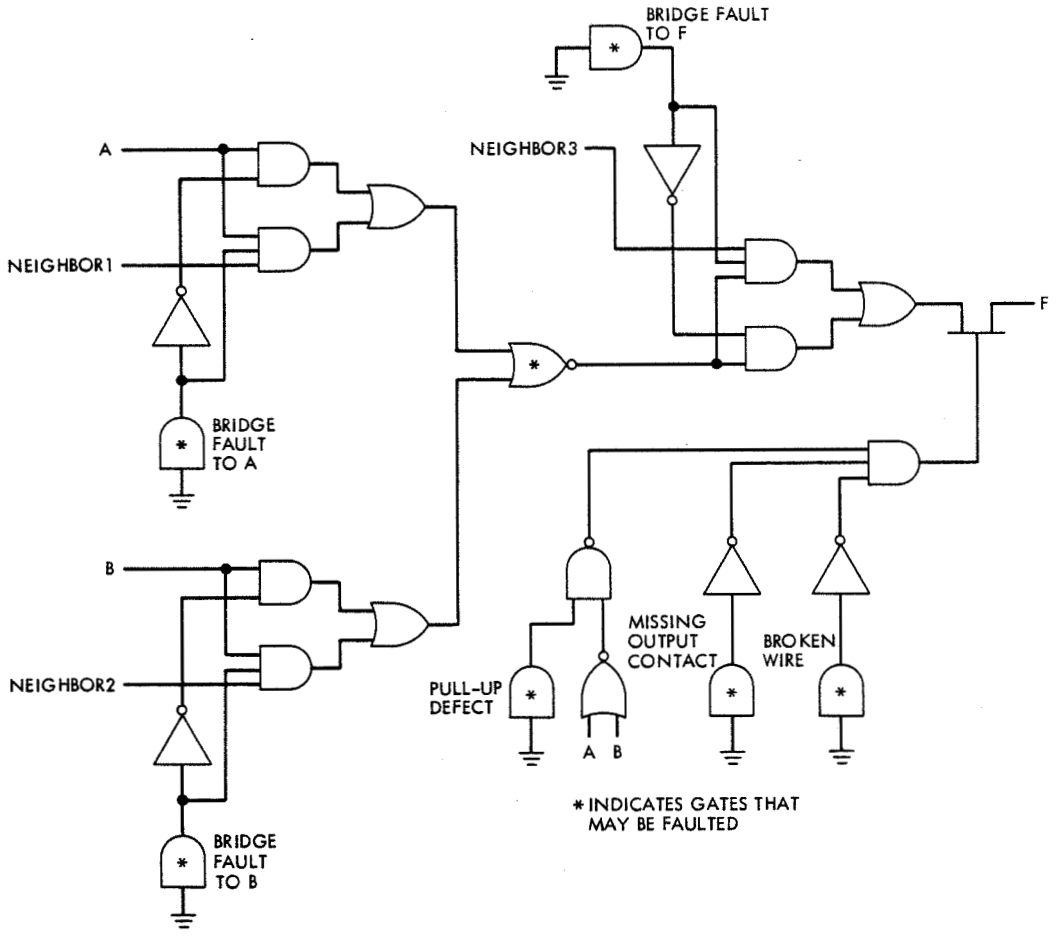

Fig. 3. NMOS failure model for a two-input NOR gate